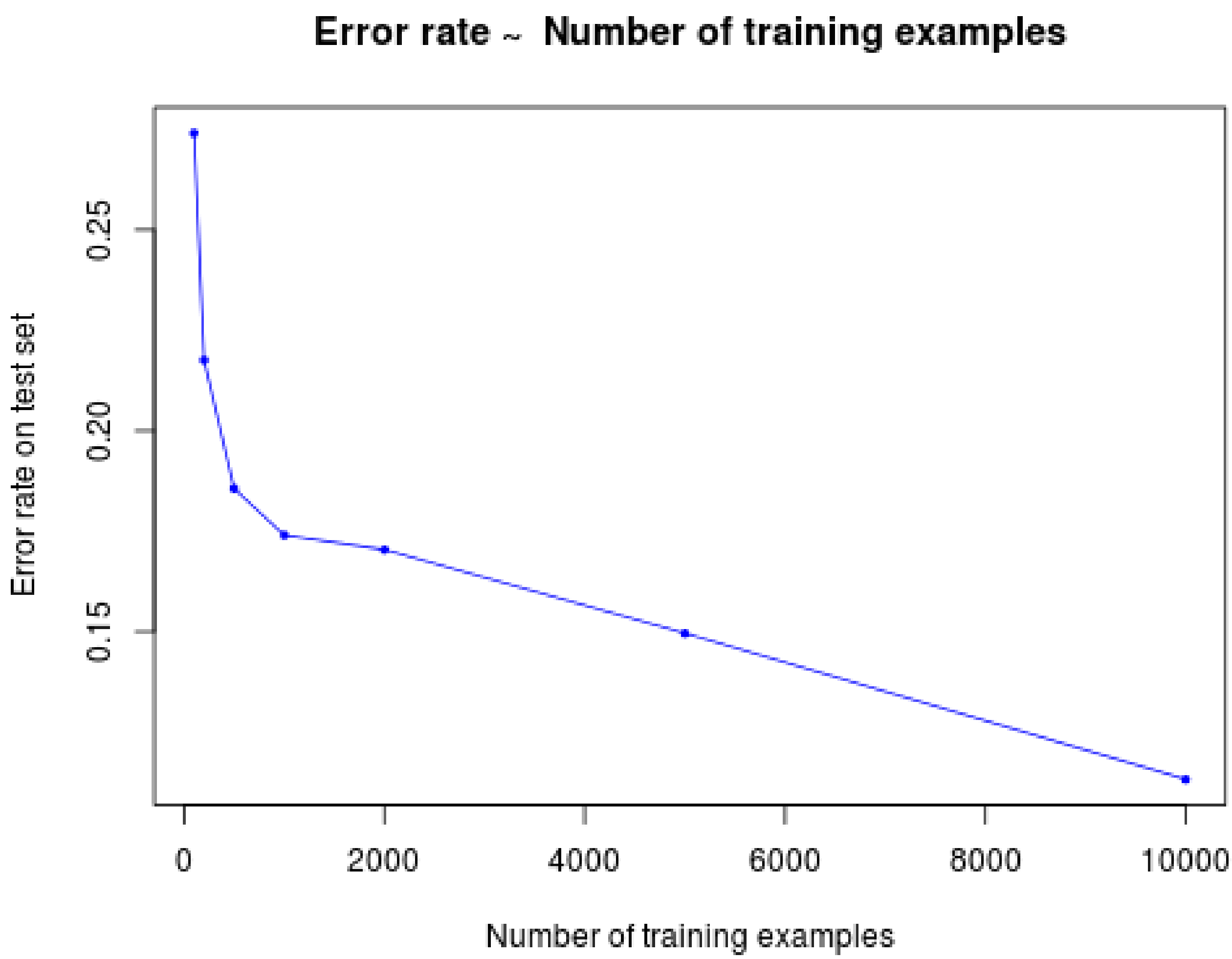
For all the questions, I used '-s 2' option for training.

problem 1:

For this question, I used the default c = 1, p = 0.1, e = 0,01, B = -1.

As seen from the plot, the error rate for the test set goes smaller as the training set becomes larger, which means that we are likely to get a better model given a larger training set.



Problem2:

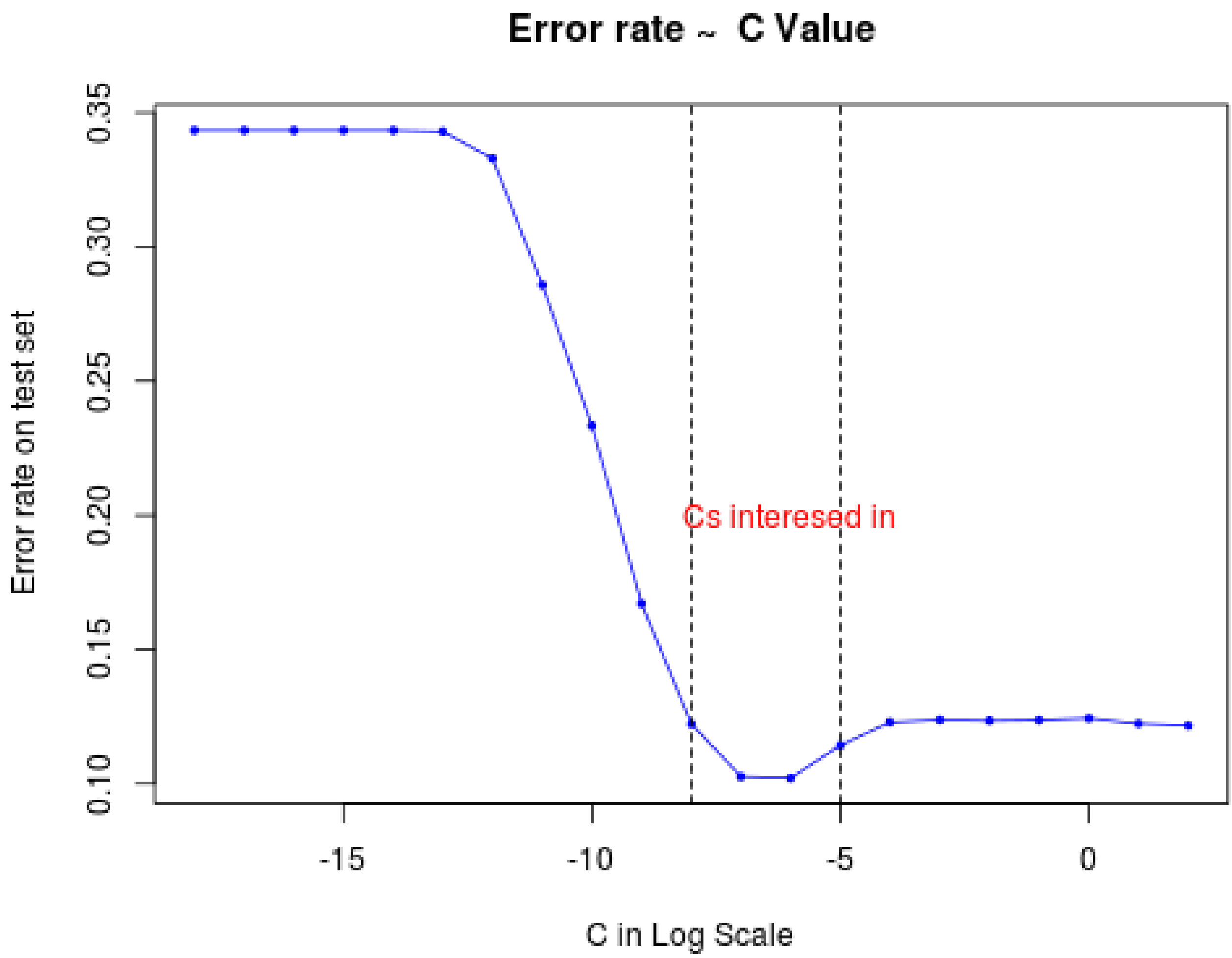
Since the actual images are almost evenly distributed among 0-9, we can tell the prediction accuary by looking at the numbers of occurence of errors rather than the frequency.

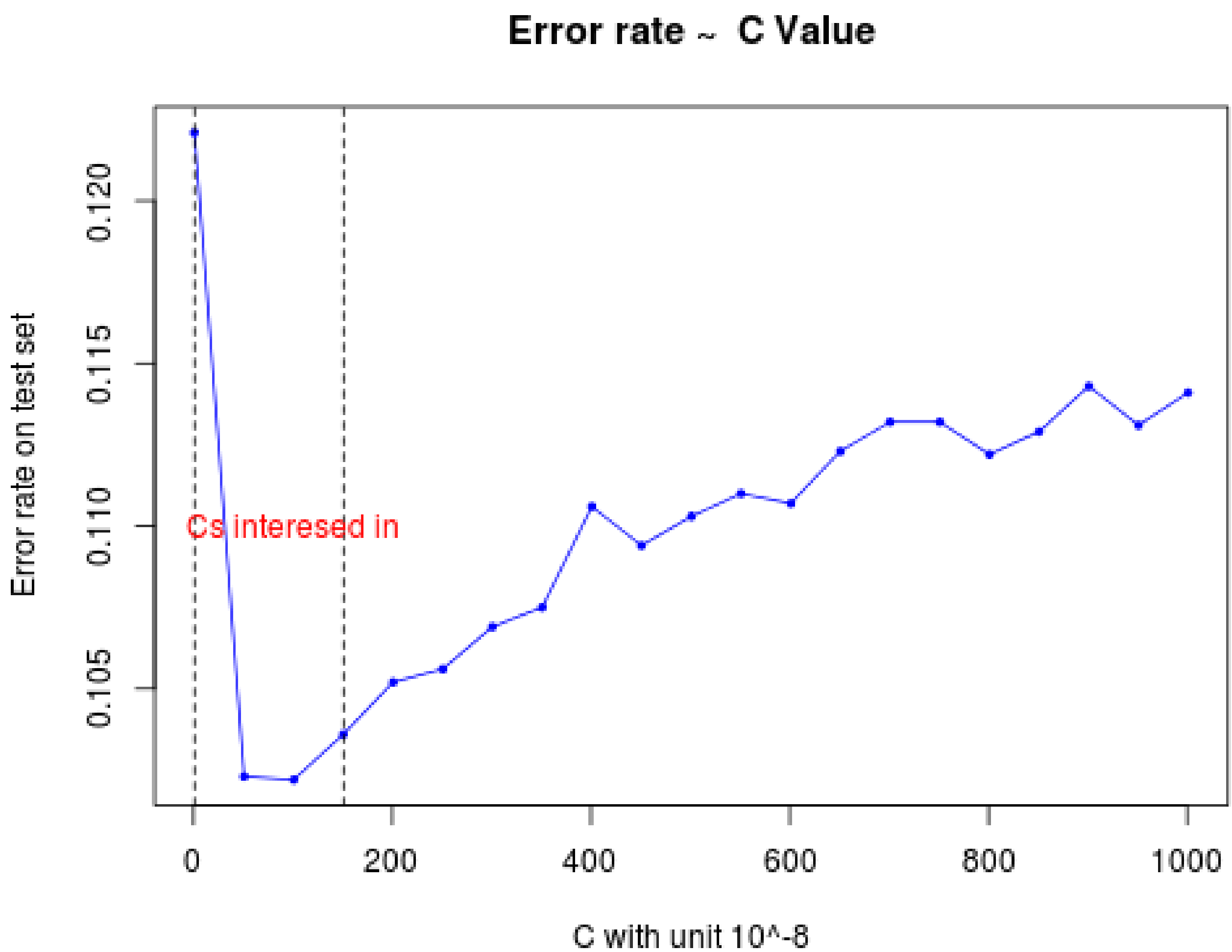
We can see:
0 and 1 are the easiest to predict with rather few errors.
2 is easy to be predicted as 8.
3 is easy to be predicted as 2, 5 or 8.
4 is easy to be predicted as 8 or 9.
5 is easy to be predicted as 3 or 8.
6 is easy to be predicted as 5.
7 is easy to be predicted as 9.
8 is easy to be predicted as 5 or 9.
9 is easy to be predicted as 4, 7 or 8.
So if we get a prediction as 0, 1 or 6, we can be pretty sure(≥ 0.99) it is correct. However if we get a 8 or 9, we are less confident about that.

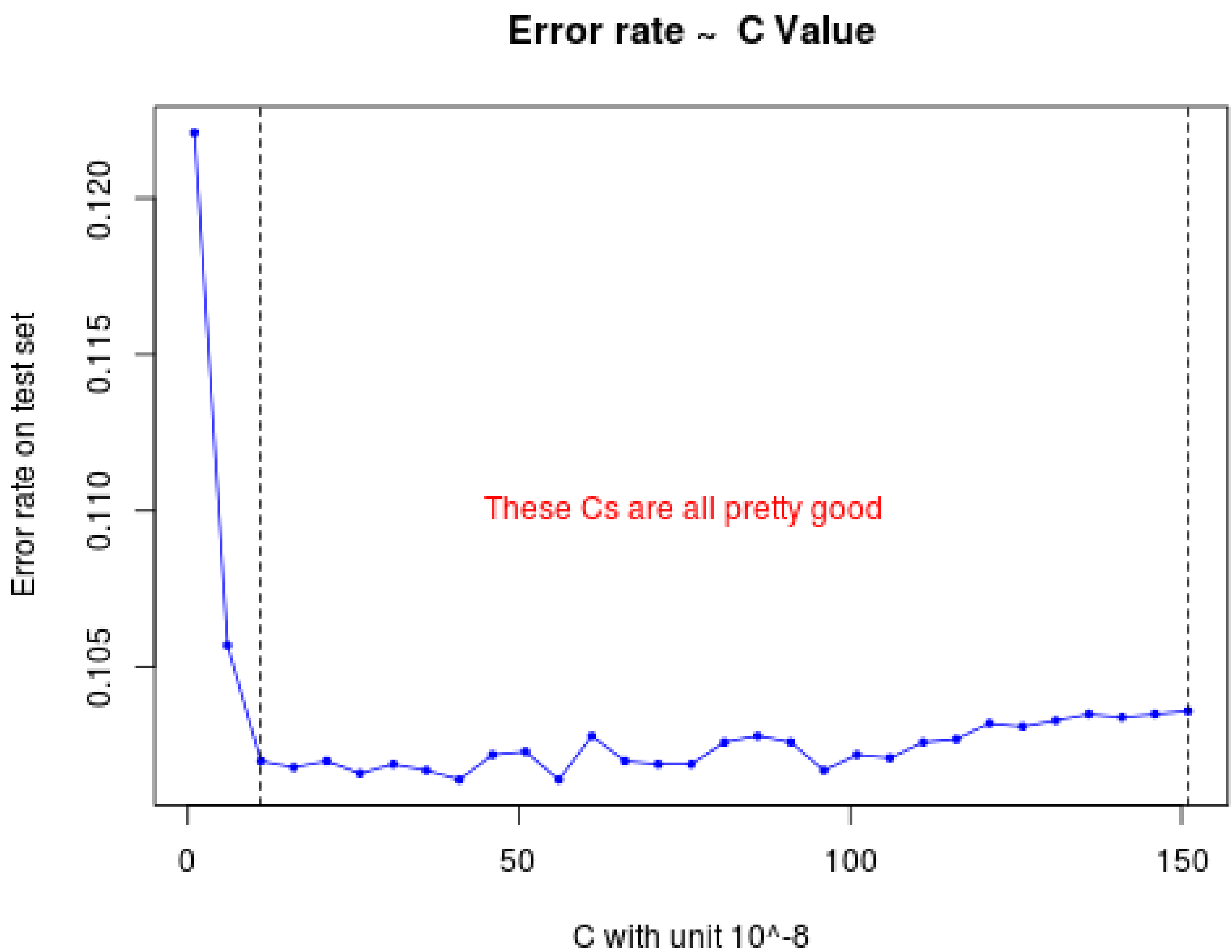
Problem 3:

Cross Validation helps us to reduce the risk of over-fitting and get a better estimation of error rate for the model. For example, when we use k-folds, we train the same parameters on slightly different training sets and test on completely different validation sets, which reduce the risk of over-fitting. Also when we use the average of k trails as the error rate, it is more accurate in term of the standard

deviation of the estimated error rate.
The best c is in range $(10^{-7}, 10^{-6})$.
Using $c = 4.2 \times 10^{-7}$, I got error rate 0.0947 for training set of size 10000, and 0.0831 for training set of size 60000.
The way to approach c in this problem:




Error rate ~ C Value
Error rate on test set
0.105
0.110
0.115
0.120
Cs interesed in
0
200
400
600
800
1000
C with unit 10^-8


Error rate ~ C Value
Error rate on test set
0.105
0.110
0.115
0.120
These Cs are all pretty good
0
50
100
150
C with unit 10^-8